**Year 12 SDD Major Work Documentation**

**By Tyler Misciagna**

**Problem Definition**

In senior school, the looming issue of the future (universities and degrees) begins to become a priority for thought and consideration by students of all academic levels. The thirteen years of mandatory education all build up to university, yet a common issue arising many students is that they are unsure of what field of study they would want to enter into. This application aims to solve the issue of students being unsure of what they want to study when they enter university, taking the student’s HSC completed subjects and using that information to provide them with a degree that is best tailored to them. It is not meant to provide the exact degree someone should do, the application is meant to be a recommendation, to guide someone to the right degree for them.

The application works by gathering the user’s HSC subjects and asking them to select their most and least favourite subjects, their best performing subject, their projected ATAR, and the university that they would like to go to. The program takes this information and filters the degrees from a csv file containing just under 2000 degrees from around 20 universities containing information such as the relevant HSC subjects and Lowest Selection Rank for each degree. This program can be used in conjunction with professional careers advisors to help speed up the initial process of determining what degrees someone should consider. A career advisor should spend more time focusing on how to apply for university and the careers which come after university rather than what type of degree someone should study.

**Legal and Ethical Considerations**

1. This program could be viewed as a replacement for professional career advisors or educational experts. Career advisors are professionals who are specifically trained to give advice regarding areas of university study and future career paths and they should be trusted above this program in terms of degree recommendation. Ideally, career advisors could use this application to assist them in deciding on professional advice.
2. The program is collecting data about a user’s subjects, ATAR and preferred university which must be collected, stored, and processed in an appropriate and transparent manner. Users are entitled to having their HSC subjects, ATAR, and preferred university private and protected from external entities and thus considerations towards user data protection must be made.
3. This program uses a variety of libraries and APIs, which if could breach terms and conditions of those libraries/APIs when uploaded publicly. This program uses Tkinter and Pandas for the GUI and table sorting respectively. If this program were to be published for commercial use, appropriate research and communications with Tkinter and Pandas must be made to negate and eliminate legal issues.
4. This program is purely to produce recommendations, not to accurately guide people’s life choices. We cannot completely guarantee that the degree output will be reliable since it uses algorithms and a set formula to output degrees. There is no possible way to perfectly tailor a degree to someone, but it is possible to lead them in the right direction.

**Functional and non-functional requirements**

|  |  |
| --- | --- |
| Functional | Non-functional |
| User enters their HSC subjects through the sub-windows which pop up when a button is pressed | Validation –  The user’s HSC subjects are checked and validated to ensure there are no discrepancies (eg. User selects too many subjects) |
| User selects their favourite subject, least favourite subject, best performing subject, their preferred university and ATAR | Error detection –  The program should be able to detect possible errors which occur and prevent them from occurring or ask the user to reinput their choices if they generate an error |
| The app should display a single degree which is perfectly (or best fitted) to the user based off their selection | User Support –  Documentation and the README file will provide the user with comprehensive instructions on how to run, understand, and operate the software |
| The app should display secondary degrees which somewhat fit the selections that the user has made. | Security –  As the program is collecting sensitive data about the user, the program should aim to have security measures to prevent data breaches |
| The app should display the degree and a hyperlink should be attached to it so that it can take you to the university website | API Compatibility –  The program should have perfect compatibility with the API and external libraries so that the program runs smoothly and efficiently |

**Storyboard**

**A screenshot of a computer

Description automatically generated**

**Context Diagram  
A diagram of a company

Description automatically generated**

**Data Flow Diagram**

A diagram of a diagram

Description automatically generated

**Structure Chart**

A white background with black text and words

Description automatically generated with medium confidence

**Algorithms**

A diagram of a flowchart

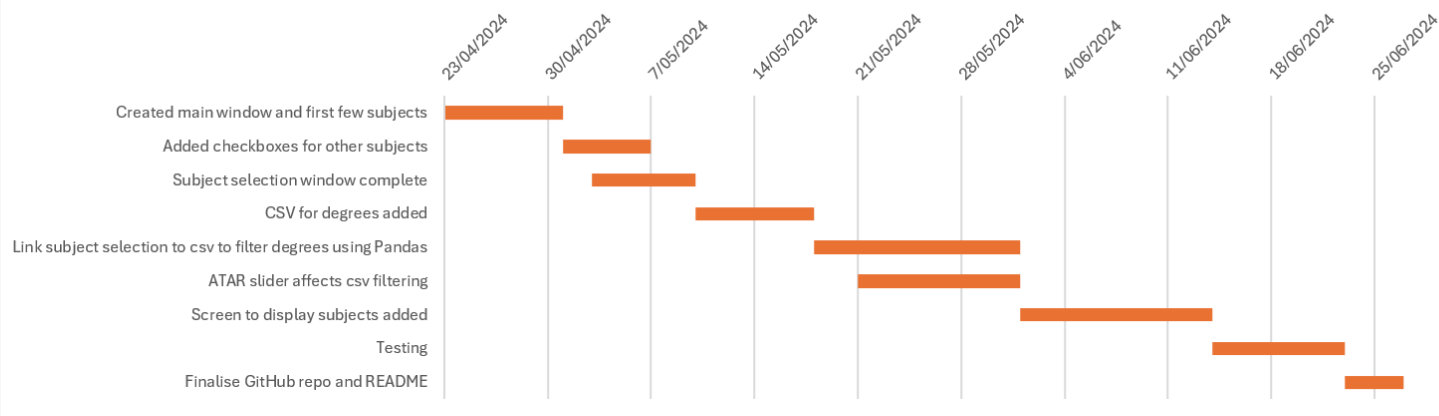
Description automatically generated

filter\_degrees() function:

1. Receives the filter categories from the user
2. Searches csv list and removes degrees from the list based off the filter categories
   1. EG. If user selects “Software” as their favourite subject, only degrees with “Software” written next to it will remain, all other degrees will be removed
3. Sorts all the new degrees into a list
4. Outputs the list as filtered\_list

**GANTT Charts**

Pre-development GANTT chart:



Post-development GANTT chart:

A graph with blue and black lines

Description automatically generated with medium confidence

My original plan was to spend less time on the main window for subject selection and more time on the screen to display subjects and the actual filtering process, as I had envisioned that it would take longer. However, I found out that the filtering process was quite straightforward and was only tedious with messing around with the Pandas API and the csv. I had also intended on testing less, but after developing I found out that a lot of potential errors came from user inputs, which meant a lot of testing was required. The initial part of creating the main window with the first subjects took longer than anticipated because I hadn’t coded for a while and needed some adjusting.

An important part of this process is that I learnt that I should allocate more time for testing on programs which require a lot of user inputs.

**Development Log**

|  |  |
| --- | --- |
| Development Log Entry 1 | |
| Date | 10/05/2024 |
| Week Number | Week 2 |
| Summary of Work Done | Finished subject selection windows. Each button (correlates to each subject category) on the root window now opens a new window where the user can select their level of English, Maths, and their other subjects.  English and Maths use radio buttons while the other categories (humanities, science, miscellaneous, technology, and languages) use checkboxes.  Some similar programs might have chosen to go for a dropdown menu but I felt that wasn’t user-friendly since you need to scroll for a long time to find your subject, and then you need to do that for each subject. |
| Challenges and Solutions | I felt like a lot of the code was unnecessary and could easily be simplified or shortened. Possibly I could use ‘for’ loops to create multiple buttons rather than producing them individually. I can implement loops for the radiobuttons but it’s harder for checkboxes because they have individual StringVar variables attached to them which have data values attached to them which I need to access individually.  I had struggled for a while to decide on the right format for the subject selection screen so it was a relief that I settled on one finally. |
| Milestones Achieved | Subject selection screen mostly complete. |
| Screenshots |  |

|  |  |
| --- | --- |
| Development Log Entry 2 | |
| Date | 18/05/2024 |
| Week Number | Week 3 |
| Summary of Work Done | * Made English and Math def functions into ‘for’ loops for simplicity * Label on root window now displays subjects selected * Made a rankings window so user can rank their subject based off enjoyment which will be used to filter the list of degrees down * Added a csv file to the repository containing a list of degrees, their names, the universities offering the degree, and suggested HSC subjects attached to each (only 209 degrees for now) * Pandas API loops through csv to filter the degrees out * Takes the favourite subject as decided by the user and removes all degrees which do not contain the favourite degree * For now, the program prints out each degree with it’s related information when the user presses print |
| Challenges and Solutions | It was a considerable struggle to get the Pandas API to work. The API is quite complicated to understand, but when I eventually worked it out, it works pretty well.  A big problem which I need to work out is that not every subject has a related degree. For example, drama is not in the csv file. So if a user selects drama as their favourite subject, it will output no degrees.  I also want more degrees, particularly from a larger range of universities. I know where to get them from but it requires scraping the text from a PDF which I’m not sure how to do. |
| Milestones Achieved | Basics of degree filtering done |
| Screenshots | A screen shot of a computer program  Description automatically generated |

|  |  |
| --- | --- |
| Development Log Entry 3 | |
| Date | 21/05/2024 |
| Week Number | Week 4 |
| Summary of Work Done | * Added error messages for invalid actions such as selecting no subjects and submitting, choosing more than one science, and choosing a beginners and continuers language. Still need to edit some of the other subjects to add more error messages, mainly for duplicate submissions. * Changed the csv. file. Line 22 had "Mathematics Adv" instead of "Mathematics Advanced" and similar on line 206. Important to change these because in the code, it is written as "Mathematics Advanced" and thus other variations won't be detected * Added a projected/estimated ATAR slide bar. Has no function yet but will add ATAR requirements for each degree soon * Made the main window disappear every time you selected a subject. Made it so users can't spam open millions of tabs. |
| Challenges and Solutions | Despite a lot of changes to prevent errors from occurring, there are still a lot of areas for errors. For example a user can put one subject in every rank. Eg. Rank 1 is Biology, Rank 2 is Biology, etc.  This isn’t too difficult to manage, just requires time and diligence.  Same challenges as last development log. I will be working on expanding the csv and ranking system. |
| Milestones Achieved | Made the program a little less vulnerable to user error  Setup the future of the program (ATAR slider) |
| Screenshots |  |

|  |  |
| --- | --- |
| Development Log Entry 4 | |
| Date | 28/05/2024 |
| Week Number | Week 5 |
| Summary of Work Done | * New CSV file added containing just under 2000 degrees from around 20 universities in New South Wales/ACT * CSV file also contains LSR (lowest selection rank) for most degrees. This is the lowest ATAR required for people to enter the subject and will act as another filtering option * Abandoned ranking system of subjects since subjects that aren’t ranked 1st are disregarded anyway. Adopted a system where the program asks for the favourite, least favourite, and best performing subject to act as the filtering options * ATAR slider is now operational and will actively filter out subjects based off ATAR. EG. If you think you will get a 75.00 ATAR, all subjects with a higher LSR will be disregarded from final results |
| Challenges and Solutions | I always envisioned that only one degree would pop up at the end of the entire process, but it appears that isn’t quite possible with the sheer number of degrees on the CSV. I can limit it down to around 15 degrees (depending on subject selection) so all I need to do is add a screen to display the degrees.  It was a challenge to get the ATAR and LSR discriminator working. A lot of the degrees scraped from the UAC pdf didn’t have the LSR category, so I had to edit the CSV to account for those. I kept getting an error which said that it couldn’t process some universities degrees because it was dealing with floats rather than strings, so I had to change all the LSRs to floats by adding digits to the end.  To do: make the program look nicer and find errors and fix them (one big hurdle to overcome is sometimes the filter options are too specific so no degrees are outputted) |
| Milestones Achieved | The basics of the program now work. Technically the application is finished (i.e.. Its initial purpose is fulfilled) but I still want to work on it more. |
| Screenshots |  |

|  |  |
| --- | --- |
| Development Log Entry 5 | |
| Date | 26/06/2024 |
| Week Number | 9 |
| Summary of Work Done | * Finished developing completely * Final screen for displaying subjects added * Background colours added * Buttons look more coherent and colourful * Font increased for readability * Function to remove subjects already selected added for every category/sub-window * General bugfixes (E.g. Some universities in the university list did not have any degrees in the csv) * Finished commenting |
| Challenges and Solutions | The function to remove subjects became quite a difficulty, particularly for the technology category for some reason. I spent longer than I had intended to fix this, which I did by creating a separate array in which the user’s subjects would be stored.  An error that occurred often was that sometimes the degree filters were too specific and no degrees would appear. I circumvented this by progressively removing filter options until degrees would pop up. This way no matter what subjects you put in, a degree *should* appear. |
| Milestones Achieved | Program complete! |
| Screenshots |  |

**Testing Table**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test ID | Category | Test Case Description | Input to Provide | Expected Output | Actual Output | Pass/  Fail | Action Taken |
| 1 | Missing Data | Not submitting one of the filter fields (e.g. favourite subject) | “” | The entire list of degrees not filtered | The entire list of degrees not filtered | Fail | Added a statement after the submit() function is called which detects if the fields are empty |
| 2 | Abnormal Data | Inputting the same subject for least and most favourite subject | Favourite Subject: “Software”  Least Favourite Subject: “Software” | Degrees outputted | Degrees outputted | Pass | Note: there was a check already in place to check if there weren’t enough suitable degrees for the criteria, thus removing this issue |
| 3 | Abnormal Data | Inputting the same subject at the subject selection screen | Subject 1: “Software”  Subject 2: “Software” | Error | No error occurs, but there are more subjects than usual | Pass | It looks weird so I added a check if the user has already selected the chosen subject |
| 4 | Missing Data | Not inputting any subject and clicking submit | “” | No error, but user can’t choose anything under the dropdown list | No error, but user can’t choose anything under the dropdown list | Fail | Added a check to see if the user hasn’t chosen a subject and prevent them from moving forward |
| 5 | Missing Data | Not inputting a projected ATAR | “” | No degrees are outputted since it defaults to 0 | It may output degrees, but unlikely. Only a few degrees don’t have a LSR | Fail | Added a check to see if the user has not selected a projected ATAR |
| 6 | Abnormal Data | Choosing a language subject - Csv contains a subject called “languages” which generalises the three languages we have at school | “Japanese Beginners” | No degrees are outputted since the “Japanese Beginners” isn’t present in the csv (Pandas will actively look for “Japanese Beginners”) | No degrees are outputted since the “Japanese Beginners” isn’t present in the csv (Pandas will actively look for “Japanese Beginners”) | Fail | Detects if a languages subject is selected after the ranking is done and automatically changes it to “languages”. |
| 7 | Abnormal Data | Very specific data set that may not perfectly suit a degree | Favourite Subject: “PDHPE”  Least Favourite: “Mathematics”  Best Performing Subject: “Software”  ATAR: “65”  University: “Australian Catholic University” | Limited degrees outputted, between 1-10. | Limited degrees outputted, between 1-10. | Pass | Note: there was already a check created so that if no degrees are found within a criteria, it removes one of the filtering options. If nothing is found again it keeps removing filter options until degrees are found. |
| 8 | Path Coverage | Testing ATAR slider | ATAR: “80.50” | All degrees under or equal to 80.50 LSR shows | Error, can’t compare floats with strings | Fail | Made all degrees with no LSRs 0.00 rather than “” |
| 9 | Abnormal Data | Testing subjects with colloquial names or abbreviated names | “Timber” or “Software” or “DT” | If chosen as one of the filtering options, nothing will appear | Nothing appears | Fail | Standardised across CSV and code one way of describing a subject.  EG. “Software” is “Software Design and Development” |
| 10 | Abnormal Data | Inputting the same data for favourite subject and best performing subject | Favourite Subject: “Software”  Best Performing Subject:  “Software” | Degrees outputted for software | Degrees outputted for software | Pass | N/A |

**GitHub Repository**

<https://github.com/TylerMisc/Career-Advisor-Major-Work>

**Project Reflection**

My project is a career advisor application which aimed to assist career advisors and students figure out what degree they would apply for after they’ve finished their high school journey. It wasn’t meant to be a complete replacement for the career advisor, but rather a tool that they can use to fast-track the initial process of determining what degree they would be the right candidate for.

The parts of the project that were successful was the filtering of the degrees and collating the degrees from UAC and applying relevant information to those degrees such as the relevant HSC subjects, Lowest Selection Rank, and related university. The process of filtering the degrees I had imagined taking a lot longer, but it didn’t take too long to figure out the process.

There was a surprising difficulty with the subject selection because there are a lot of gimmicks involved with selecting HSC subjects, such as requiring a user to select English or preventing them from selecting more than three sciences. Another big hurdle was figuring out how to deal with very specific user requests which then didn’t produce any degrees because the program filtered out every degree. This took quite a long time and prevented me from trying to extend the scope of the program, such as potentially including links for each subject.

Another hurdle I faced was trying to make the program look user-friendly. The program uses Tkinter, which on its own looks bland and vanilla. Another angle I could have adopted is to research alternatives such as PySimpleGUI, which I attempted to implement late in the development journey, but it was harder than previously anticipated because much of the code already implemented had been tailored to the Tkinter widgets and screens, making it harder to convert over. Because the cross-over was so late into development, I ended up leaving it in Tkinter.

I had initially planned to generate one degree that was the best for the user and then a list of possible secondary degrees which somewhat fit the user. This was not really possible because of the way my program is run. It doesn’t calculate a percentage fit, but rather limits the list of degrees outputted based off of filtering options, so I ended up not including this.

Some key takeaways that I learnt as a software engineer is to talk with other people on how the program should look before coding. This is extremely important because it’s a lot easier to implement drastic changes before you begin coding rather than after. It could have also helped make the program flow nicer for the user and help with the flow of logic.